词法分析

2020年5月8日 19:51

1. 正则表达式Regular Expression及其表示的语言regular language/set
   a. ε：L(ε)={ε}
   b. a(a属于字母表Σ)：L(a)={a}
   c. 当r和s都是RE时以下表达式也是RE（运算优先级：克林闭包>连接>或）
      i. r|s：L(r|s) = L(r)∪L(s)（或运算）
      ii. rs：L(rs) = L(r) L(s)（连接）
      iii. r*：L(r*)= (L(r))*（克林闭包）
      iv. (r)：L((r)) = L(r)
   d. 正则文法和正则表达式可互相表达
2. 正则定义regular expression
   a. 正则定义：给某些RE命名的定义序列
   b. 形如d→r（d是字母表没有的新符号，r是正则表达式，之后可用d代替r）
3. 有穷自动机Finite Automata
      i. 由两位神经物理学家 MeCuloch和Pitts于1948年首先提出，是对一类处理系统建立的数学模型
      ii. 具有一系列离散的输入输出信息和有穷数目的内部状态
      iii. 系统只需要根据当前所处的状态和当前面临的输入信息就可以决定系统的后继行为。每当系统处理了当前的输入后，系统的内部状态也将发生改变
   a. 转换图transition graph：由start剪头指向表示初态（只能有一个），双圈表示终态（可有多个）
   b. 最长子串匹配原则Longest String Matching Principle：当输入串的多个前缀与一个或多个模式匹配时，总是选择最长的前缀进行匹配（到达某个终态之后，只要输入带上还有符号， DFA就继续前进，以便寻找尽可能长的匹配）
   c. 有穷自动机和正则表达式也可以互相转化，即正则文法 ⇔ 正则表达式 ⇔ FA
4. 确定的FA (Deterministic finite automata, DFA)M = (S，Σ，δ，s0，F)
   a. S：有穷状态集
   b. Σ：输入字母表，即输入符号集合。一般假设ε不是 Σ中的元素
   c. δ：将S×Σ映射到**S**的转换函数。对任意s∈S, a∈Σ, δ(s,a)表示 从状态s出发，沿着标记为a的边所能到达的**状态**
   d. s0：开始状态 (或初始状态)，s0∈ S
   e. F：接收状态（或终止状态）集合，F⊆ S
   f. DFA算法伪码

```
s = s0 ;
while （（c=nextChar（）） ! = eof ）
        s = move ( s , c ) ; // 从状态s出发，沿着标记为c的边所能到达的状态
if (s在接受状态集F中) return“yes”；
else return “no”；
```

5. 非确定的FA (Nondeterministic finite automata, NFA)
    a. δ：将S×Σ映射到**2^S**的转换函数。对任意s∈S, a∈Σ, δ(s,a)表示 从状态s出发，沿着标记为a的边所能到达的**状态的集合**
    b. 其他同上
    c. DFA和NFA也是可以互相转化的
    d. NFA易于理解，难于计算机实现
    e. 可以通过加入空边（ε-边）使NFA容易理解
6. 从RE到DFA
    a. 从RE到NFA
        i. 将连接运算画成一组邻边
        ii. 将克林闭包运算画成自环
        iii. 将或运算画成平行边
    b. 从NFA到DFA（subset construction子集构造法）
        i. 将可达状态集合视作一个单独的新状态（空边闭包？）
        ii. 新状态对各个输出的后续状态各原状态的输出状态的并集状态
7. 子集构造法伪码
    a. ε-closure (s)：能够从NFA的状态s开始只通过ε转换到达的NFA状态集合
    b. ε-closure (T)：能够从T 中的某个NFA状态s开始只通过ε转换到达的NFA状态集合,即U_{s ∈ T} ε-closure (s)
    c. move(T , a)：能够从T中的某个状态s出发通过标号为a的转换到达的NFA状态的集合

```
// 一开始，ε-closure （ s0 ） 是Dstates 中的唯一状态，且它未加标记
while（在Dstates中有一个未标记状态T ） {
        给T加上标记;
        for（每个输入符号a） {
                U = ε-closure(move(T, a));
                if ( U不在Dstates中) 将U加入到Dstates中，且不加标记； Dtran[T, a]=U ;
        }
}
```

    d. 空闭包伪码：

```
将T的所有状态压入stack中;
将ε-closure （T ）初始化为T ;
while（stack非空） {
        将栈顶元素t 给弹出栈中;
        for（每个满足如下条件的u ： 从t出发有一个标号为ε的转换到达状态u)
                if ( u不在ε-closure（T ） 中） {
                        将u加入到ε-closure（T ）中;
                        将u压入栈中;
                }
}
```

8. 识别单词的DFA
    a. 若能进入终态，说明是该语言的单词
    b. 若转换表中的信息为空，则进入错误处理
    c. 错误处理：若之前有终态字符，则将该终态视为单词尾，从下一个字符开始重新识别；若找不到，则进入错误恢复
    d. 错误恢复：有很多种，如panic mode恐慌模式：从剩余的输入中不断删除，知道能从剩余输入的开头发现正确字符